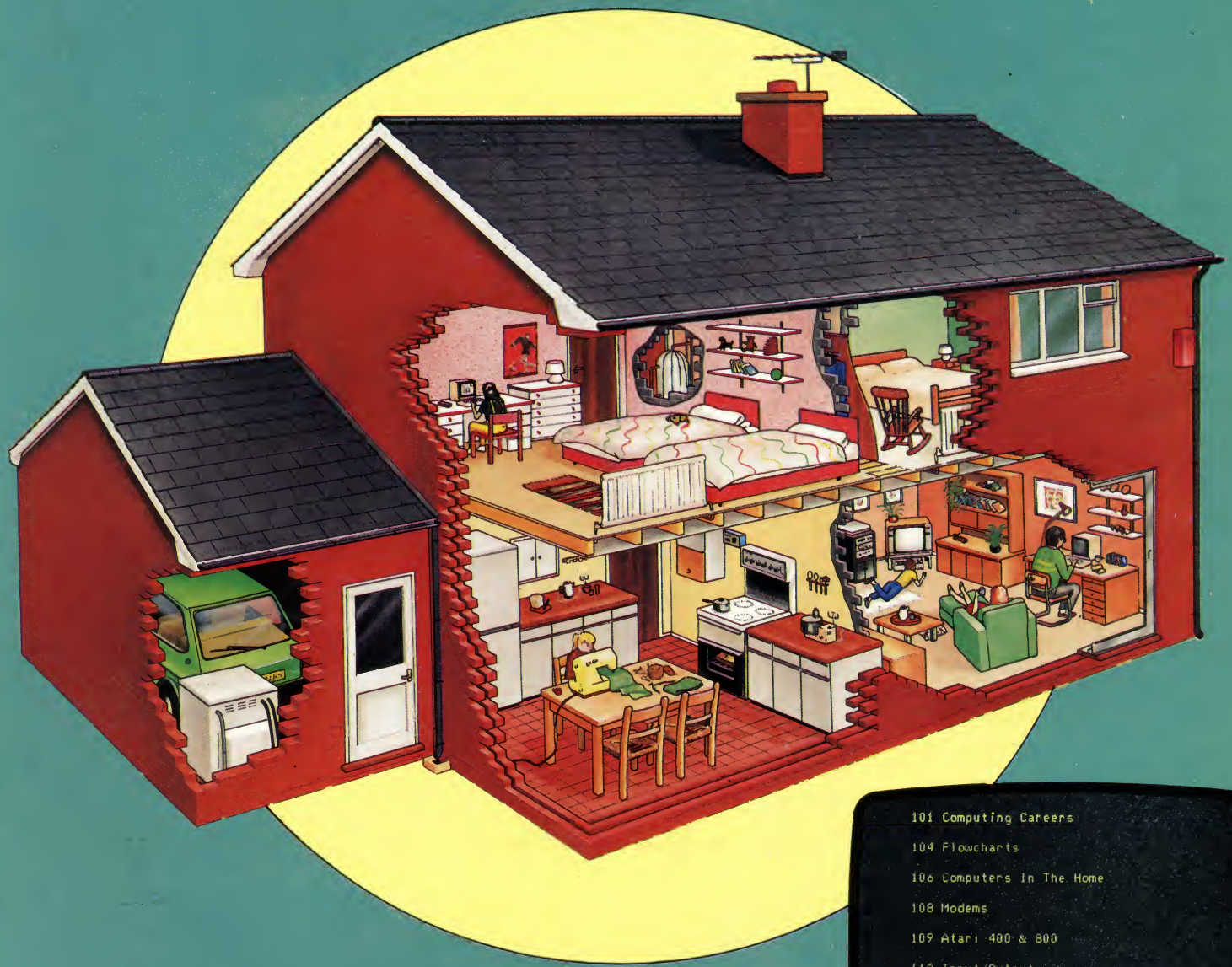
ISSN 0265-2919

R 80p

THE HOME COMPUTER COURSE 6

MASTERING YOUR HOME COMPUTER IN 24 WEEKS



101 Computing Careers
104 Flowcharts
106 Computers In The Home
108 Modems
109 Atari 400 & 800
112 Input/Output
114 Disk Drives
116 Basic Programming
119 How Computers Multiply
120 Pioneers In Computing, Part 1

An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Hardware

Domestic Science You may be surprised to learn how many household appliances contain a microprocessor — 106

Atari 400 & 800 These micros are leaders in the games market — 109

Software

Charting The Course The best way of planning well-structured programs — 104

Two's Company We discover how computers perform multiplication — 119

Insights

Computing Careers Entering the professional world of computing — 101

The Missing Link Modems allow micros to communicate over telephone lines — 108

Flat Spin Disk drives offer high-speed information retrieval and storage — 114

Basic Programming

Braving The Elements We introduce subscripted variables — 116

Passwords To Computing

Digital Dialogue The two-way flow of information between the computer and its peripherals — 112

Pioneers In Computing

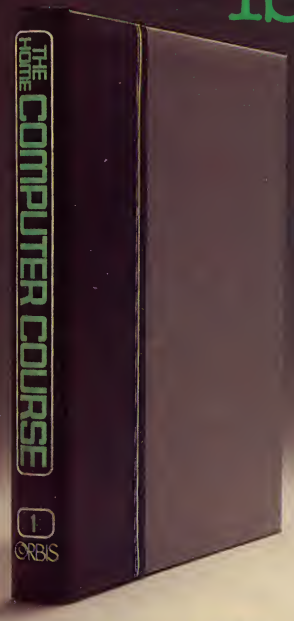
Sir Clive Sinclair The man who has made computers accessible to almost everyone — 120

Home Computer News A review of personal computers as seen at London's Barbican Centre — INSIDE BACK COVER

Next Week

- **We look at the Dragon 32, a home computer successfully marketed by the Welsh Development Agency, which offers good facilities and excellent value for money**
- **From sand to circuit. We reveal the intricacies of chip manufacture**
- **In recent years the field of medicine has benefited considerably from the introduction of microprocessors**

YOUR BINDER ORDER FORM IS WITH THIS ISSUE.



Binders may be subject to import duty and/or local tax.

Overseas readers: This binder offer applies to readers in the UK, Eire and Australia only.

Editor Gareth Jefferson; **Art Editor** David Whelan; **Production Editor** Catherine Cardwell; **Picture Editor** Claudia Zeff; **Designer** Hazel Bennington; **Art Assistants** Steve Broadhurst, Julie-Anne Chambers, Liz Dixon; **Sub Editor** Teresa Bushell; **Researcher** Melanie Davis; **Consultant Editor** David Tebbutt; **Project Manager** Jimmy Egerton; **Contributors** Tim Heath, Roger Ford, Richard King, Henry Budgett, Nick Walsh, Ray Hammond, Garry Marshall, Rom Smith; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd;** **© 1983 by Orbis Publishing Ltd; Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

How to obtain your copies of HOME COMPUTER COURSE – Copies are obtainable by placing a regular order at your newsagent.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 4, 5 and 6. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595. Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

Computing Careers

The computer professional acquires skill by working initially as a technician, and progressing up through the ranks

COURTESY OF ICL

Gentle Giants
Large commercial computers like this one (known as 'mainframes' to differentiate them from mini- and microcomputers), require a team of highly trained operators to keep them running at peak efficiency. Machines of this size are capable of running hundreds of programs simultaneously, and serving thousands of users anywhere in the world by means of telephone lines, microwave links and communications satellites. A computer room will often contain a number of such machines, each communicating with the other

The increasing use of computers at home and in schools is producing many gifted programmers — people who might otherwise never have considered the possibility of a career in computing. But the harsh truth is that, as always, a little learning is a dangerous thing — especially, it appears, if that little learning is of the BASIC language.

It is important to understand that the requirements of a professional programmer are fundamentally different from those of a home user, and that many of the attributes are not transferable.

For the school-leaver with a profound interest in computers, a college course on the subject, or direct career entry to computing, seems an obvious choice. Many colleges and universities offer degree courses with a computer qualification at the end and successful students are likely to find themselves able to choose from a variety of job offers. Unemployment in the computing industry has been limited to lower-level computer staff — largely programmers and operators — and the demand for engineers, systems analysts and designers continues unabated.

One option becoming increasingly available is to teach computing at school. Until now, computing as a subject in its own right has been the preserve of universities and colleges. Education is desperately short of trained computer personnel, and such a career would undoubtedly be very rewarding.

There are perhaps six main levels of hierarchy in the computer industry. The lowest grade may be described as 'skilled user.' This category includes workers who have learned to operate computers in particular tasks, such as word-processing or accountancy. Often these skills are picked up as a sub-set of skills to other occupations — e.g. secretarial or office administration — but they also include computer industry functions such as terminal operator, card-punch (data) operator and the like. These jobs require a basic set of school or college qualifications, and the ability to think clearly. Skills such as keyboard operation are normally taught on the job.

Next step up is the computer operator. Though the computers used in industry are quite different in appearance and feel from home computers, they are based on the same principles, so some familiarity is useful. Operators soon come to understand the fundamentals of how computers work, and so becoming an operator is a good springboard to becoming a programmer. Bear in mind, though, that the work can be quite demanding physically. Most large installations, for example, are in operation for 168 hours a week, and need to be manned for all that time.

To become a programmer, the main attributes one needs include a clear, methodical mind and an ability to concentrate on minute detail. It takes a very special type of aptitude to make a skilled programmer, and while normal entry qualifications are a degree or senior school-leaving exam passes, natural ability to work logically often counts for more. Programmers do enter the industry without formal qualifications, and it is this opening that attracts many parents, hopeful for their sons' or daughters' programming abilities.

The Choice For A Lifetime

Like any other part of a modern corporation, the computer department is organised along hierarchical lines. At its head is the **Data Processing Manager**, who is responsible for all the many and varied tasks that fall under the main heading of information processing.

All computer professionals are firstly technicians, and acquire management skills as they progress up through the ranks. The three main areas of specialisation are computer operations, programming, and systems analysis, and there is an element of mobility between specialisations in the promotion path.

In common with the other professions, it is worth entering the field as well qualified as possible. While it may not appear to make too much difference at the beginning, a less qualified person will soon find the path barred. It's much more difficult to get a university degree while doing a full time job! Additionally, organisations like the British Computer Society now offer professionally-recognised qualifications, usually by examination, and for an aspiring programmer or analyst these are a good indicator of standing within the industry.

Analysts

Before starting on any job, it is as well to look closely at the objectives and the resources available. The **Systems Analyst** has the task of interviewing users, to determine their needs, to match these needs with resources, and suggest a method of solving the problem. In order to evolve a system of working for other people, the Analyst must be a logical thinker with good communications skills and a spark of creativity. He is often the DP department's salesperson, so must always make a favourable impression on his 'customers' — the computer users in the company.

Programmers

The **Programmer** takes the broad strategy worked out by the Analyst and converts it, first into a tactical plan, breaking the job down into manageable segments, and then into code that the computer can recognise and interpret. **Applications Programmers** are concerned with writing programs to do specific jobs, while **Systems Programmers** are more involved with the overall performance of the data processing system. Applications Programmers tend to work in isolation, even though they may be part of a project team. For them, the ability to concentrate attention on the task in hand is really important. Systems Programmers need that too, but also a calm outlook. 'If you can keep your head when all about you are losing theirs . . .' then perhaps you have the makings of a Systems Programmer.

Development Engineers

Though the time may come when computers themselves develop the new generation of machines, it's in the brain of the **Development Engineer** that this process of innovation takes place now. The development engineer is part scientist, part technician. It is his job to take advantage of new discoveries and theoretical developments to improve and enhance the performance of a given piece of equipment. Doctorates abound in this field where even the least well qualified is likely to have spent five or more years at university.

TONY SLEEP

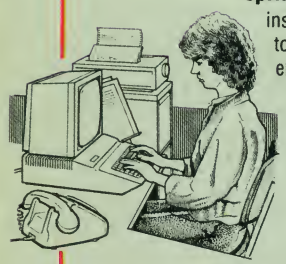
Field Engineers

Often, the only chance an Operator has to relax is when the computer develops a fault, and a **Field Engineer** has to be called in to fix it. Given the modern computer's ability to diagnose its own failings, and the almost universal adoption of modular construction, the engineer's job has become somewhat simplified, but a field engineer must still be competent in digital electronics. He must also be a skilled mechanic, capable of working to finer tolerances than the average watchmaker. To enter the field, a degree level qualification is usually required.

Operators

Physically, the most demanding of all jobs in the industry is operating a large multi- programming/multi-user computer.

But as well as walking miles in a shift with disk packs, tapes or boxes of paper, the operator must be fully conversant with the computer's operating system, and with the relative importance of the jobs being run on the machine at any one time. A **Senior Operator** will be called on to make decisions affecting the work of many other parts of the company's business by allowing or denying access to the computer system.



Operators in smaller installations are often called on to help programmers and engineers diagnose faults, as well as simply running the job at hand. Most important, though, is a thorough knowledge of the program's operating method. 'User-friendly' software makes the operator's job easier, and a well prompted' program can be run by relatively inexperienced staff with little loss of efficiency.

KAI CHOI

Least demanding of all, intellectually, is the **Data Entry Operator's** job. The skills required here are much the same as those needed by a copy typist — speed and accuracy. At worst the task is boring and repetitious, but in many small installations this is offset by the opportunity to become involved in other aspects of the computer department's activities.

COURTESY OF THE SUNDAY TIMES

As we mentioned earlier, an understanding of BASIC is not necessarily an open door to the computing industry. Although it is a popular language on home computers, most professionals regard it as badly structured and consider that it encourages bad programming habits and sloppy thinking. This is a real problem as most children with home computing experience are likely to have learned BASIC rather than one of the better structured languages such as LOGO or COMAL.

Several universities and colleges that run courses in computing now express a preference for entrants *who have not learned* BASIC as they feel that the language forms habits that are hard to break.

Despite this problem, many youngsters are finding a way to make their BASIC programming skills pay off. Many are writing games in BASIC that appeal to other youngsters, and software companies are anxious to get hold of games that are so directly targetted to an adolescent mind. Some of the 'whizz kids' featured in newspapers as earning vast sums at a young age write only in BASIC, and have no real understanding of computing. Others are genuine phenomena, writing in Assembly Language (the low-level language that controls a microprocessor's machine code very efficiently) and set for a very bright future indeed. Newspaper reporters are seldom qualified to distinguish between the two, and such reports can lead parents to think that their computer-obsessed offspring is ready to enter the big money. It is possible, but unlikely.

Inside The Business

In the computer industry proper, programmers are split into two groups: applications programmers and systems programmers. Applications programmers write programs to carry out a specific task. Systems programmers are 'housekeepers', writing programs to keep the computer system in order — to detect faults, for example. The applications programmer is likely to meet people outside the computer room — clients — and is likely to work as part of a team developing programs for a specific task. The systems programmers are more specialist, and tend to work alone. They are talking directly to the machine's 'intelligence'.

But at this point, the computer industry draws an artificial dividing line, beyond which it denies access to all but the brightest programmers and the best-qualified university graduates. This is the realm that belongs to the systems analysts and designers.

Systems analysts consider a problem and then decide how a computer can help to solve it. For example: an oil company discovers a new deposit under the sea bed. They have measured the extent of the deposit and found that the quality of oil varies widely. The oil company has to decide whether or not to invest the billions of dollars necessary to exploit the oil field. This decision will be based on projections about the state of the international oil market for the life of the field (say 20 years) and the company must decide which part of the field to drill first. Because the investment is so vast, the oil company hands the problem to its computer people for analysis. The analyst considers the problem, consults economists, oil marketing experts, geologists and other specialists and over a long period of time constructs a computer 'model' of the oil field.

The oil company executives can then play 'what if?' with this model, discovering how various decisions about price, refining techniques and market approaches would affect overall performance, and they are provided with all the information they need to make their final decisions about how best to exploit their field.

There are several other important roles in the computer industry, although few are as highly regarded as the systems analyst. Perhaps the exception is hardware design. There are openings for electronics engineers at all levels from high street repair centres to research departments, but the areas of product development and pure research are only open to those with the highest electronic engineering qualifications.

Many analysts and designers of both machines and software move on to managerial and consultancy positions, but these titles often indicate only that the individual is working in a more powerful role, very often self-employed. The work content of the job often remains the same.

On any typical day there is a massive shortfall in skilled computer personnel — some put it as high as 20,000 or more in this country alone — and at the same time a huge pool of unemployed, many of them graduates of universities and polytechnics. This obvious skills mis-match is a source of worry to educationalists and industrialists alike, and serious steps are being taken to rectify the situation, including re-training programmes for those qualified in other fields and a much wider variety of opportunities to learn at primary, secondary and tertiary levels.

Several governments, Britain's in particular, consider that microelectronics may provide an answer to some of the short-term unemployment problems. The Youth Training Scheme, which aims to provide 'on the job' training and work experience for unemployed school leavers, now offers 4,500 places at Information Technology Centres in Britain. At these centres, young school leavers learn about various aspects of microcomputing while receiving a training allowance equal to unemployment benefit. Other projects within the scheme offer some computer familiarisation to those who fell through the net at school (either because they left school before the computer arrived, or because they weren't 'selected' to use it) and also improve their prospects of finding a job, because for those who leave school without any computer familiarity or literacy, employment prospects can seem grim.

David Simmonds

David Simmonds, 17, earned himself £10,000 during his summer holidays.

He's a programming wizard who writes programs for Commodore (makers of the PET and Vic computers). Unlike many teenage boys, David writes 'serious' software that has commercial applications and he is expecting to find a lucrative niche in the computer industry when he has finished studying.

David started playing with a computer his father brought home from work, but he quickly abandoned game-playing and got down to discovering how to program. Initially David had some of his programs published in Commodore's user magazine and slowly but surely he began to sell copies of his programs, for a few pounds

Commodore eventually took notice and David persuaded them to let him show them what he could do. The result was his first serious programming assignment

Eugene Evans

Eugene Evans is 17 years old and his earnings are reported to be £40,000 a year!

Eugene is one of the many whizz kids now springing up in computer programming and he is helping to keep his employers, Imagine Software of Liverpool, among the top computer game producers in the country.

The high earnings made by these programming wizards usually take the form of royalties on the sales of games (rather like authors' book royalties) and teenage boys are best suited to develop games that will appeal to other teenage boys — the main market for computer games

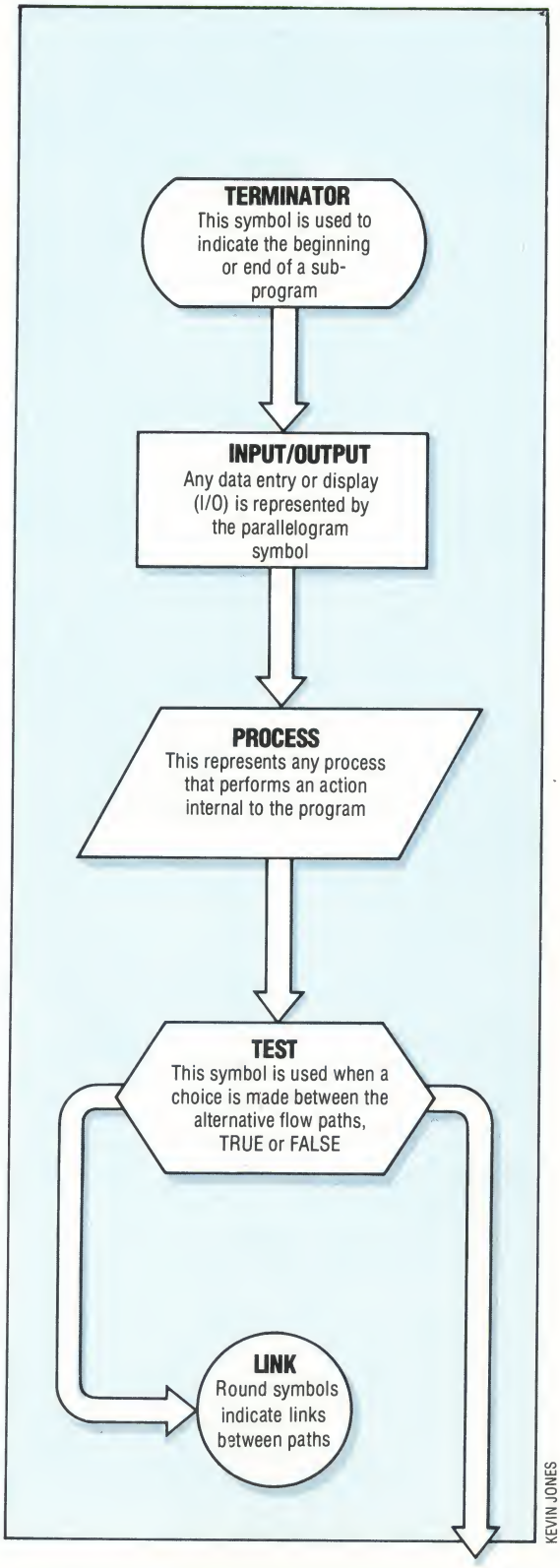
Charting The Course

The conscientious use of flow diagrams leads to efficient and well-organised programs

The Flow Of Information
The real purpose of a flow diagram is to indicate in a simple, concise manner, the flow of information and control through a computer program.

Most important are the 'test' points, where control passes to a point other than the next in sequence. A simple graphical representation of this passage of control is much easier to grasp than a similar statement written out — a picture really can be worth a thousand words!

The 'TEST' symbol can be represented by either a flattened hexagon, as shown here, or by an elongated diamond shape

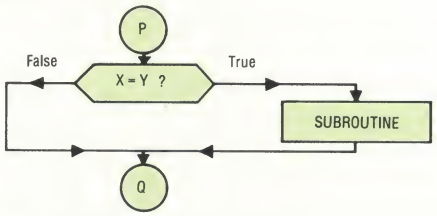

TERMINATOR
This symbol is used to indicate the beginning or end of a sub-program
INPUT/OUTPUT
Any data entry or display (I/O) is represented by the parallelogram symbol
PROCESS
This represents any process that performs an action internal to the program
TEST
This symbol is used when a choice is made between the alternative flow paths, TRUE or FALSE
LINK
Round symbols indicate links between paths

KEVIN JONES

A problem can be represented in a simple pictorial way by drawing diagrams to show the steps required in processing, and the flow paths or routes connecting them. These 'flowcharts' are useful as a means of understanding a problem, and in working out its solution.

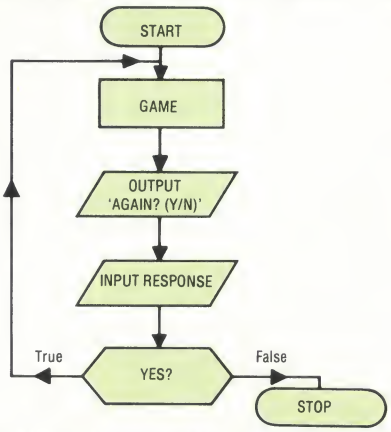
Each box symbol in a flowchart represents a process or action, and the lines that connect these action boxes depict the possible paths through them. 'Traffic flow' is one-way, so arrows are used to indicate direction, which is normally top to bottom, and left to right across the diagram.

Whenever a choice is to be made, a hexagonal or diamond shaped 'decision box' is used. Control flows in by one path, as before, but may pass out in one of two directions, depending on the result of the test in question. If the test is to determine whether a single process is to be performed or not, then only one of the exit paths will contain a 'process box'. Here is an example of a test to decide whether or not to branch to a sub-routine:




```
120 IF X = Y THEN GOSUB 300
```

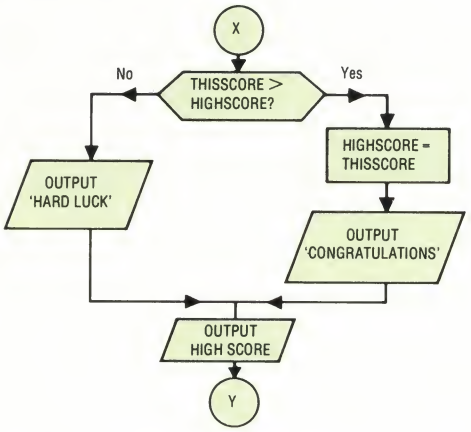
The decision box is also used to indicate the test that terminates a loop. In the example given below, control is returned to the start of a program if there is a positive reply to the question 'AGAIN?':




```
90 REM**START OF GAME**
100
800
```

```
810 PRINT "AGAIN? (Y/N)";
820 INPUT R$
830 IF R$ = "Y" THEN GOTO 100
840 END
```

We may wish to make a decision that will result in one of two distinctly different courses of action being followed. In the example shown below, we compare a player's game score to the highest previous score:




```
1200 IF THISSCORE > HIGHSCORE THEN GOTO
     1230
1210 PRINT "HARD LUCK. YOU HAVE TO BEAT";
1220 GOTO 1250
1230 LET HIGHSCORE = THISSCORE
1240 PRINT "CONGRATULATIONS! A NEW HIGH
     OF";
1250 PRINT HIGHSCORE
```

Note that the value of HIGHSCORE is printed in both events, and that the two possible flow paths rejoin in the process to become the single entry to this output operation.

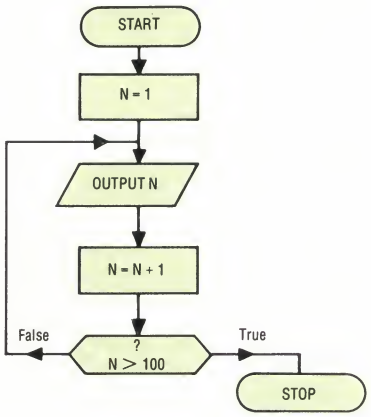
All decisions are taken as a result of tests similar to this, which deliver a positive or negative, a True or a False result. As you can see, this purely binary decision-making process denies the possibility of a 'maybe' answer. You can use whatever terms you wish, but don't forget to label the two exit paths accordingly!

All programming languages have an inherent decision statement which, if the True condition is satisfied, cause a conditional branch, but which drops control through to the next statement if the result is False. In the case of a dialect of BASIC that allows only a simple IF–THEN, we must mimic the conditional branch by means of a GOTO statement, as in line 1200 of the last example. The statement in line 1210 will only be executed if the result of the test in line 1200 is False.

But what about the second use of GOTO in line 1220? As you can see, the use of GOTO at the end of the test, to solve the problem of the destination of the conditional branch, has forced us to use this method to 'join up' the two possible control paths again, in this case at line 1250.

The use of flowcharts usually encourages the introduction of GOTOs as a means of following the point to point graphical representation of the program. In general, this use of unconditional jumps is rather dangerous. If the version of BASIC that is being used forces this solution, then a flow diagram is an excellent method of assessing the way in which control passes out of the program's normal succession.

Let's use one last example to examine how the use of a flowchart allows us to represent accurately the necessary steps to perform a simple task: printing out all the numbers between one and one hundred.




```
10 LET N = 1
20 PRINT N
30 LET N = N + 1
40 IF N > 100 THEN END
50 GOTO 20
```

The use of flow diagrams in this way tends to encourage a step by step approach to program writing which, especially in larger projects, often leads to a rather inelegant result. For those with even a passing knowledge of the BASIC language, the use of a FOR–NEXT loop is obviously indicated. For example:

```
10 FOR N = 1 TO 100
20 PRINT N
30 NEXT N
40 END
```

The flowchart is incapable of representing this piece of BASIC 'shorthand', and to follow it exactly would lead one to a less efficient way of solving the problem. It does, however, give us some information on the structure of the FOR–NEXT loop, and so is of value when we come to examine this and other BASIC functions, to determine how they are constructed.

Flow diagrams are particularly useful during the planning or conceptualising stage of programming, especially in the 'tricky' parts. Experienced programmers tend to use them less than beginners, and will often resort to a flow diagram to illustrate and document a piece of software written without their aid. But whether a flow diagram is drawn out on paper, or it simply exists inside the programmer's subconscious mind, the concept of charting the flow of information and control is central to the use of computers as a problem-solving tool.

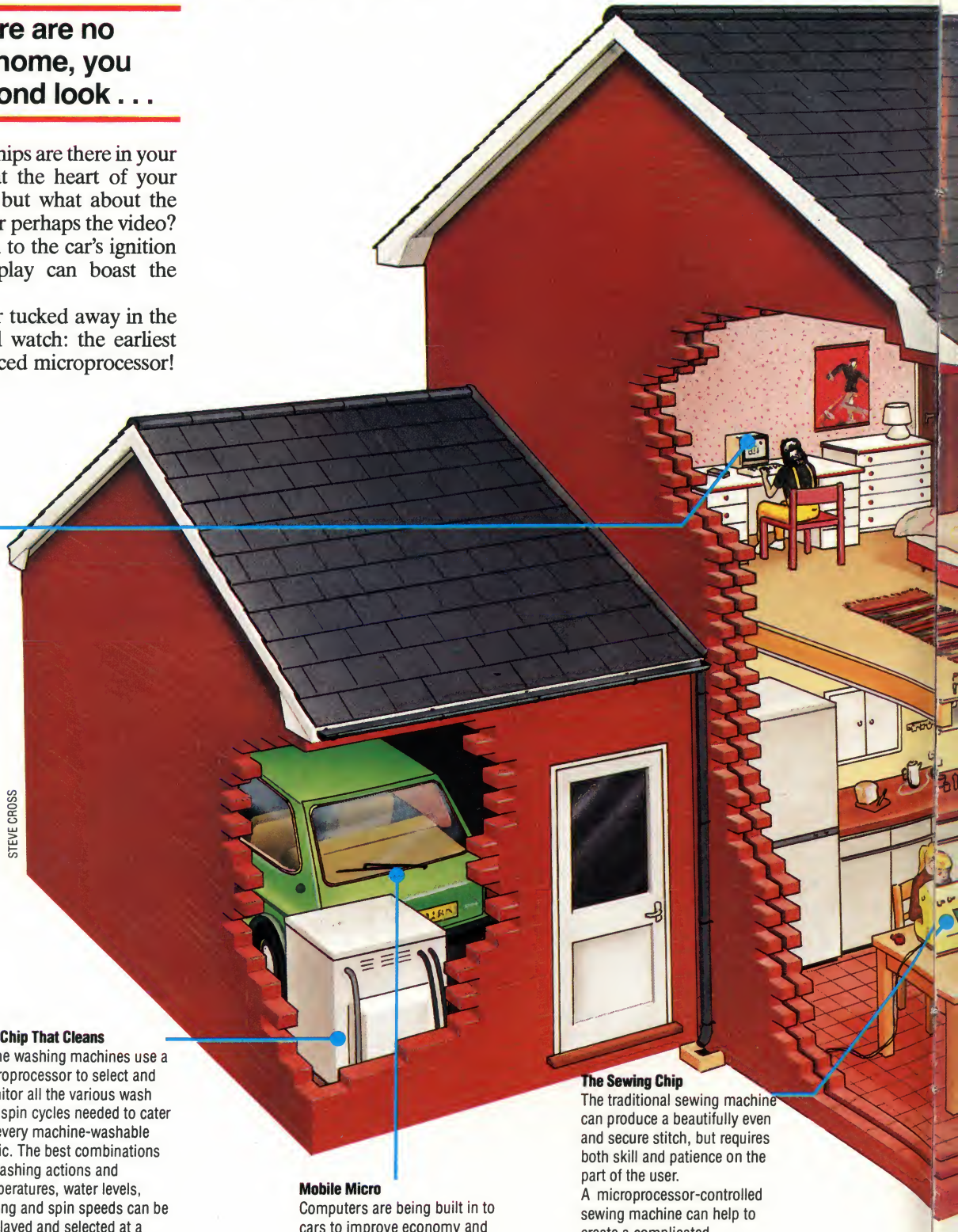
Domestic Science

If you think that there are no computers in your home, you ought to take a second look . . .

How many microprocessor chips are there in your home? There will be one at the heart of your home computer, of course, but what about the washing machine, the hi-fi, or perhaps the video?

Everything from the oven to the car's ignition control and dashboard display can boast the presence of a chip.

Don't forget the calculator tucked away in the desk drawer, or your digital watch: the earliest examples of the mass-produced microprocessor!



STEVE CROSS

Child's Play
Children often make fuller use of computers in the home than adults, accepting them as naturally as the television set. Knowing about the turtle and LOGO teaches children to explore and learn by themselves. Even a simple LOGO that only has turtle graphics can help the younger child at home

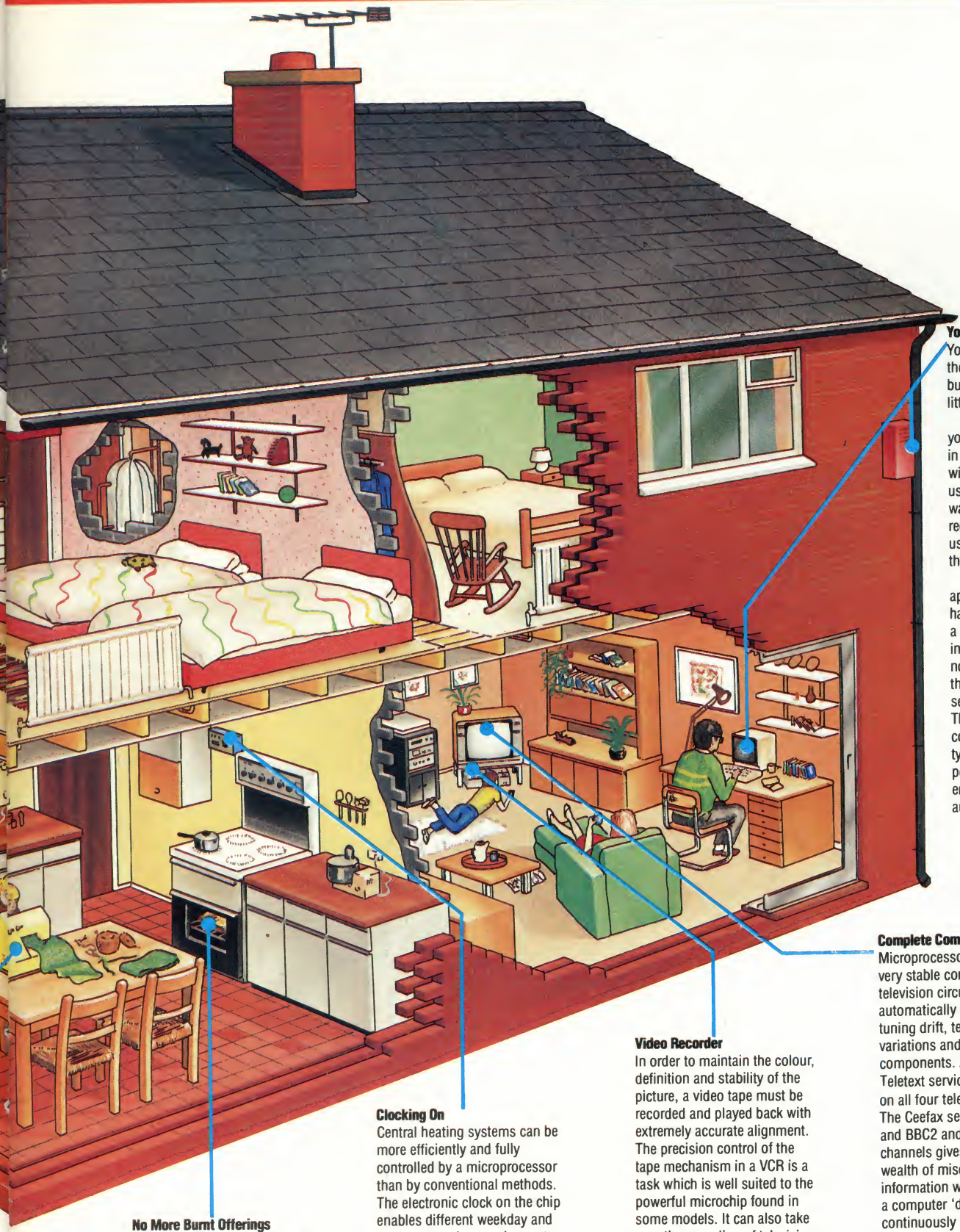
Good quality educational software is also available (see page 81). Games can stimulate and educate, but often the interests and talents of a child are best developed and encouraged by exposure to the problems encountered in actually programming a computer.

LOGO is now becoming widely and cheaply available for many home computers, and offers enormous potential for encouraging the best approach to problem solving in many fields other than computer programming

The Chip That Cleans
Some washing machines use a microprocessor to select and monitor all the various wash and spin cycles needed to cater for every machine-washable fabric. The best combinations of washing actions and temperatures, water levels, rinsing and spin speeds can be displayed and selected at a touch. Because there are no moving parts, except for the drum of course, the life of the machine will also be much longer and servicing costs considerably reduced

Mobile Micro
Computers are being built in to cars to improve economy and provide novel functions. They can calculate fuel consumption, monitor speed, act as burglar alarms and even speak to the driver, warning of low oil pressure or battery power

The Sewing Chip
The traditional sewing machine can produce a beautifully even and secure stitch, but requires both skill and patience on the part of the user.
A microprocessor-controlled sewing machine can help to create a complicated embroidery pattern or an awkward stitch with little effort. Besides the selection of pre-set stitches, these sewing machines can be programmed to produce other stitches that the built-in memory will store, even when the machine is switched off



Your Home Computer
You may originally have used the computer to play games, but how about something just a little more challenging?

If you run a business, or if you are a parent with an interest in your child's education, you will probably already be making use of the computer in many ways. You could be keeping records of all your accounts, or using the educational value of the home computer.

But there are a host of other applications that may never have occurred to you. If you are a DIY enthusiast, you may be interested to note that there is no limit to the sophistication that you could build into a security system, for example. The computer can monitor concealed detectors of many types and initiate alarms, perhaps even dialling up emergency services automatically

Complete Compensation
Microprocessors can provide very stable control of the colour television circuitry. They automatically compensate for tuning drift, temperature variations and ageing of components. Also, of course, Teletext services are broadcast on all four television channels. The Ceefax service on BBC1 and BBC2 and Oracle on ITV channels gives you access to a wealth of miscellaneous information which is stored on a computer 'data base'. This is continuously updated so that the latest developments in news, weather, sport and even stocks and shares are instantly available at any time

Video Recorder
In order to maintain the colour, definition and stability of the picture, a video tape must be recorded and played back with extremely accurate alignment. The precision control of the tape mechanism in a VCR is a task which is well suited to the powerful microchip found in some models. It can also take over the recording of television programs while you're out. Just program the automatic tuner for the times and channels you want

Clocking On
Central heating systems can be more efficiently and fully controlled by a microprocessor than by conventional methods. The electronic clock on the chip enables different weekday and weekend heating requirements to be programmed appropriately. Separate areas such as the bedrooms and the greenhouse or garage can also have their own programs for timing and temperature control. Control such as this saves energy and cuts the cost of electricity bills

No More Burnt Offerings
A computerised oven can help to produce perfectly cooked dishes by accurate timing and temperature control. While the meal cooks to its programmed temperature and time, the latest recipe is on the screen via the home computer's Teletext adaptor

The Missing Link

Information can be passed from one computer to another over thousands of miles by means of the modem

IAN DOBBIE

Acoustic Coupler
Most modems in use are 'all electronic' devices that connect directly to the telephone lines, plugging into the telephone socket. Telephone companies set very rigid standards for devices such as modems. This tends to make them expensive. A cheaper solution, since it by-passes the regulations, is to use an 'acoustic coupler'. This is a type of modem that converts the sine-wave audio-frequency signals into actual sounds fed into a small loudspeaker

The term 'modem' is a contraction of 'modulator/demodulator'. Although modems have been commercially available for about five years, they are now being used by owners of home computers in increasing numbers. If we can all have a computer of our own, what's the point of spending money on a modem to link it to the telephone system?

Modems allow your computer to 'talk' to other computers all over the world. The only requirement is for the computer at the other end of the telephone line to have its own modem. This other computer may be just an ordinary home micro owned by another enthusiast, or it could just as easily be a huge mainframe owned by a university or financial institution. Connecting your computer to a large mainframe can give access to large databanks, information services and even the latest stock market prices. Connecting your micro to a friend's enables you to exchange software or send inexpensive 'electronic' mail and even play two-way games.

Modems work in a similar way to the cassette interface supplied with most home computers. Both cassette interfaces and modems convert the computer's ones and zeros into audio frequencies. In the case of cassette interfaces, these frequencies can easily be recorded as though they were audio signals on the cassette tape. With modems, the audio frequencies are simply sent down the telephone line to be converted back into binary numbers by the modem at the other end.

Cassette interfaces, however, need only to convert binary into audio signals in order to record on the tape (this process is called modulating). Or they do the opposite and convert the audio signals replayed from the cassette into binary (this is called demodulating). Most modems, on the other hand, are designed for two-way communication over a single telephone line and so they need two frequency bands and four individual frequencies. One popular standard uses a frequency of 1,070Hz for a 0 and 1,270Hz for a 1 for transmitting and 2,025Hz for a 0 and 2,225Hz for a 1 for receiving. You will notice that the two frequencies in each of the two bands (the low frequency band and the high one) are very close. There's only a 200Hz difference in frequency for a 1 and a 0 in both bands. This contrasts sharply with cassette interfaces where the frequency that represents a 1 is usually twice as high as the frequency for a 0. To be able to decode frequencies so close together calls for rather complex electronic circuitry and this tends to make modems something of a luxury — modems can cost as much as many small home micros.

FAX Machine
FAX machines (short for facsimile machines) are fast becoming popular in offices in Europe and the United States. In Japan even the smallest businesses have them and many private homes use them too. FAX machines can transmit large documents, including drawings and pictures, to other FAX machines in a matter of seconds, using nothing more than a built-in modem and an ordinary telephone

Atari 400 & 800

Game-playing is the special strength of the Atari range of computers

CHRIS STEVENS

The fortunes of Atari, now with headquarters in Sunnyvale, in California's Silicon Valley, rest on the phenomenal success of their arcade games. The first of these was 'Pong', played in black-and-white on the television screen.

From this humble beginning, Atari grew, and eventually became part of the vast Warner Communications Group, and now, some six years later, are among the largest manufacturers of home computers, as well as having a very large slice of the arcade market.

With its excellent standard of construction, and reassuringly heavy feel, the Atari range of home computers, comprising the 800 and 400 models, has set standards which others seek to emulate. Superb graphics, well-developed software and — until recently — a very high price tag all contribute to this quality image.

Selling for around £150, the 400 differs from the larger model in price (the 800 costs £280), in maximum memory size (fixed at 16 Kbytes in the 400, expandable to 48 Kbytes in the larger model), in having one instead of two cartridge ports, and, perhaps of less immediate importance, being restricted to use via a domestic television, where the 800 has the option of display via a monitor. Most obvious, and perhaps most critical, amongst the differences, however, is in the keyboard.

In order to overcome the problem of inconsistent signal levels, Atari home computers do not use domestic cassette recorders, but rather require Atari's own model. A large proportion of users however, have at least one disk drive, in order to take advantage of the wide range of software packages available on disk.

Atari Keyboards
The striking difference between these two Atari models lies in their keyboards. While the larger 800 model is equipped with a full typewriter-style keyboard the smaller model has a membrane type which, while better than some, still suffers from the in-built faults of other similar units – notably a lack of 'feel', and sometimes unpredictable response. It is as well to bear in mind, however, that very expensive 'ruggedised' machines, produced for industrial and military applications, use this method to secure against dust and the occasional spilled cup of coffee!

The Disk Drive
Generally considered to be the most useful peripheral, the Atari 810 is now beginning to show its age. At only 88 Kbytes per disk, it's rather small, and since it connects to the machine via a serial interface it's also fairly slow. However, it has a sophisticated operating system which has many features derived from other programs

The Cassette Unit
Being a special unit, designed to work with the Atari computers, the Atari 410 Cassette Unit is more reliable and easier to operate than the regular domestic variety. For the same reason, it doesn't have a speaker, which reduces size and weight, and neither can an 'ordinary' cassette be used instead. Shortcomings of the system are principally the lack of named cassette-files

The Atari Joystick
The Atari Joystick is one of the poorer add-ons for the machine. It's a switch-type, so there is no variability. It gives just a push in the required direction, and it's rather stiff, so other makers have produced alternatives

CPU BOARD

ANTIC
One of the specialised chips that give the Atari its impressive features, ANTIC controls the screen-scrolling, lightpen and one of the interrupts

CTIA or GTIA
This chip, unique to the Atari, handles colour, some miscellaneous I/O and the Player-Missile graphics

Colour Clock
Turning this control will alter the colour. The various graphic resolutions are selected by varying the speed of this clock

6502 CPU

Colour Adjustment

RAM BOARDS

The Atari can hold up to three RAM boards. In this way the memory can be expanded to 48 Kbytes

Master Clock

POKEY
The third of the custom-made chips, POKEY takes care of the keyboard, serial I/O, the system-timers and also controls the sound

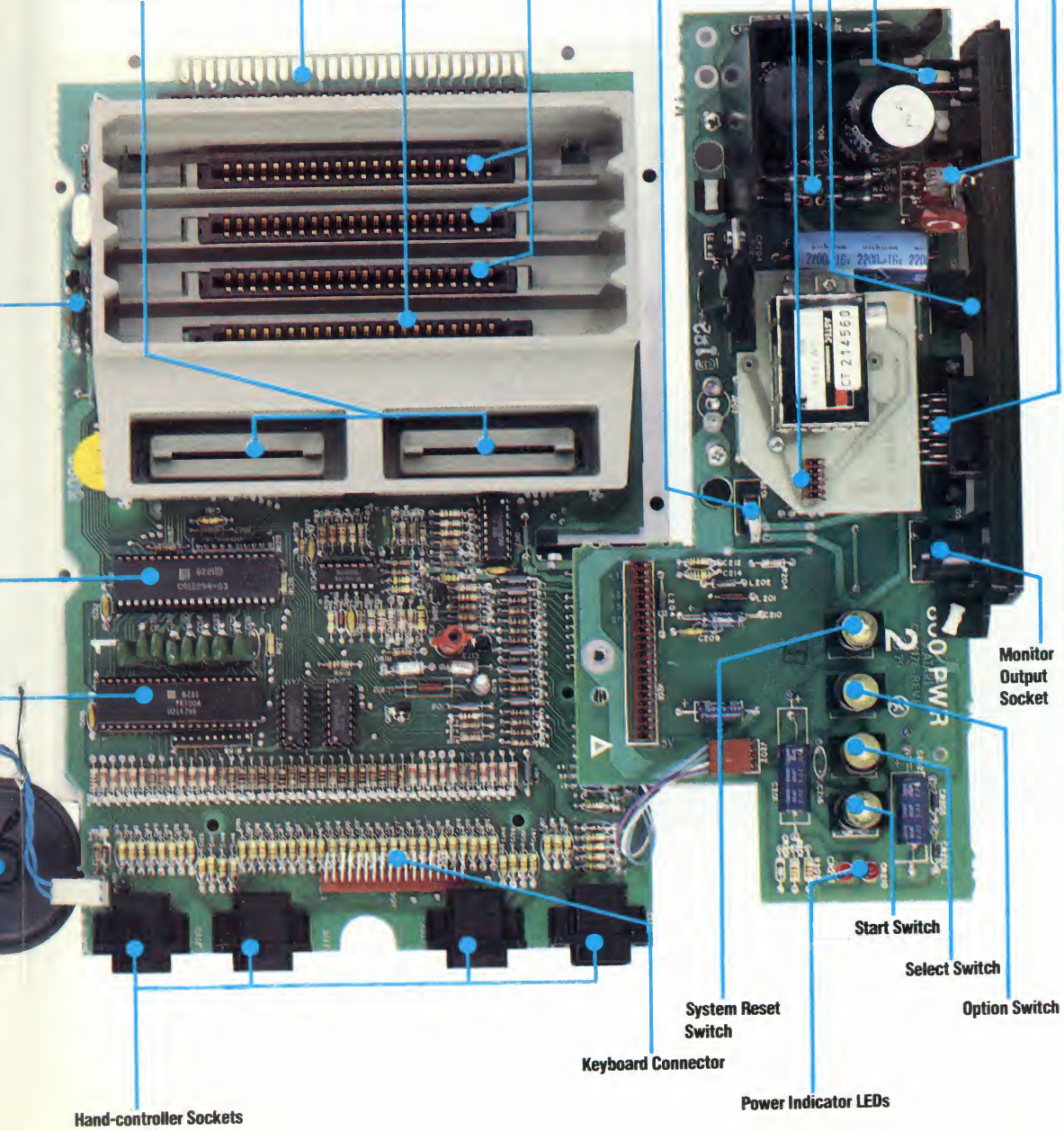
6520 Peripheral Interface Adaptor
This chip looks after the Hand Controllers

Speaker

PERSONALITY BOARD

The Atari can be made into quite a different machine by replacing these ROMs with others. For example, alternative languages could be used.

MOTHERBOARD



Rectifying Diodes
The Atari power supply gives out an AC current, so these components are used to convert to DC

Peripheral Socket

Power On/Off Switch

Expansion Slots 1, 2 and 3
A RAM cartridge can be plugged into these connectors, each adding 16K to the machine

Channel Selector

CPU Connector
Since the Atari has the CPU on a separate card, this slot is provided to hold it

Video Signal Connector

Power Input Socket

Left and Right Cartridge Slots
These each take ROM-cartridges that are pre-programmed with games or useful programs

Interlock Switch
This disconnects the power whenever the lid is raised, to reduce the danger to the user

ROM Slot

Monitor Output Socket

Start Switch

Select Switch

Option Switch

System Reset Switch

Keyboard Connector

Power Indicator LEDs

Hand-controller Sockets

ATARI 800

PRICE
£280

SIZE
405 × 330 × 110mm

WEIGHT
4,200g

CPU
6502

CLOCK SPEED
1.79MHz

MEMORY
16 K to 48 Kbytes

VIDEO DISPLAY
Text screen: 40 × 24 characters, graphics screen maximum of 329 × 192 dots including 16 colours and 8 shades

INTERFACES
TV connector, monitor, cassette recorder (dedicated unit), 4 joysticks, serial port

LANGUAGE SUPPLIED
BASIC

OTHER LANGUAGES AVAILABLE
BASIC A+, PILOT, C

COMES WITH
Power supply unit (without plug), manual

KEYBOARD
57 individual moving keys, plus 3 function keys

DOCUMENTATION
The introductory manuals are clearly written, accurate and well produced. Atari will supply full technical notes for the more experienced user. This advanced manual is exactly the same as used by Atari's engineers, and not only contains the full circuit diagram, but also listings of many of the programs that control the inside workings of the computer (the system software). The only shortcoming of the manual supplied is the format, which is in pages for a 3-ring binder that is not provided

Digital Dialogue

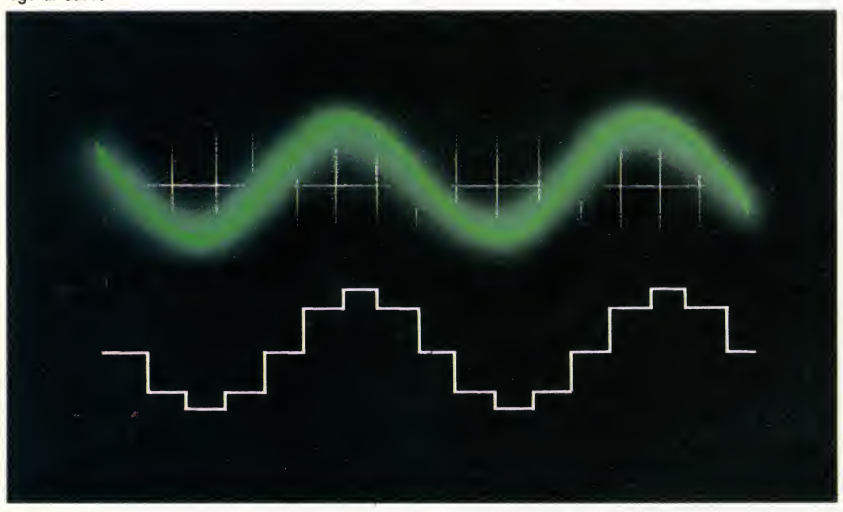
Input and Output are essential to the operation of any computer system

Analogue To Digital

In the real world, few pieces of information come in discrete, digital steps. Rather, they are infinitely variable like noise levels or the tides.

In order to make these data comprehensible to the computer, the signal must first be digitised. The Analogue-to-Digital (A/D) converter takes samples from the signal source at a known, constant rate — perhaps one hundred every second. Each of these samples is stored in a separate memory location as a digital value, thus allowing calculations of variance to be made, and out-of-limits conditions to be recognised.

Digital-to-Analogue (D/A) converters perform a similar function in reverse, statistical techniques being used to smooth out the peaks into a regular curve



MARK WATKINSON

Input/Output, or I/O as it is commonly abbreviated, is the term used to describe the transfer of information between the CPU (the Central Processing Unit forming the heart of the computer) and the 'outside world'. The 'outside world' in this context means any devices that may be connected to the computer. It does not include RAM and ROM memory, which are considered to be integral to the computer. The distinction between what is held to be 'inside' the computer and 'outside' it is somewhat arbitrary. But all of the logic circuits (see page 92), designed to work in close conjunction with the CPU and main memory, are considered to be part of the 'inside' of the computer.

External devices, which use I/O for communication with the computer, include a wide variety of peripherals ranging from the keyboard to floppy disk drives, joysticks, printers and video display units.

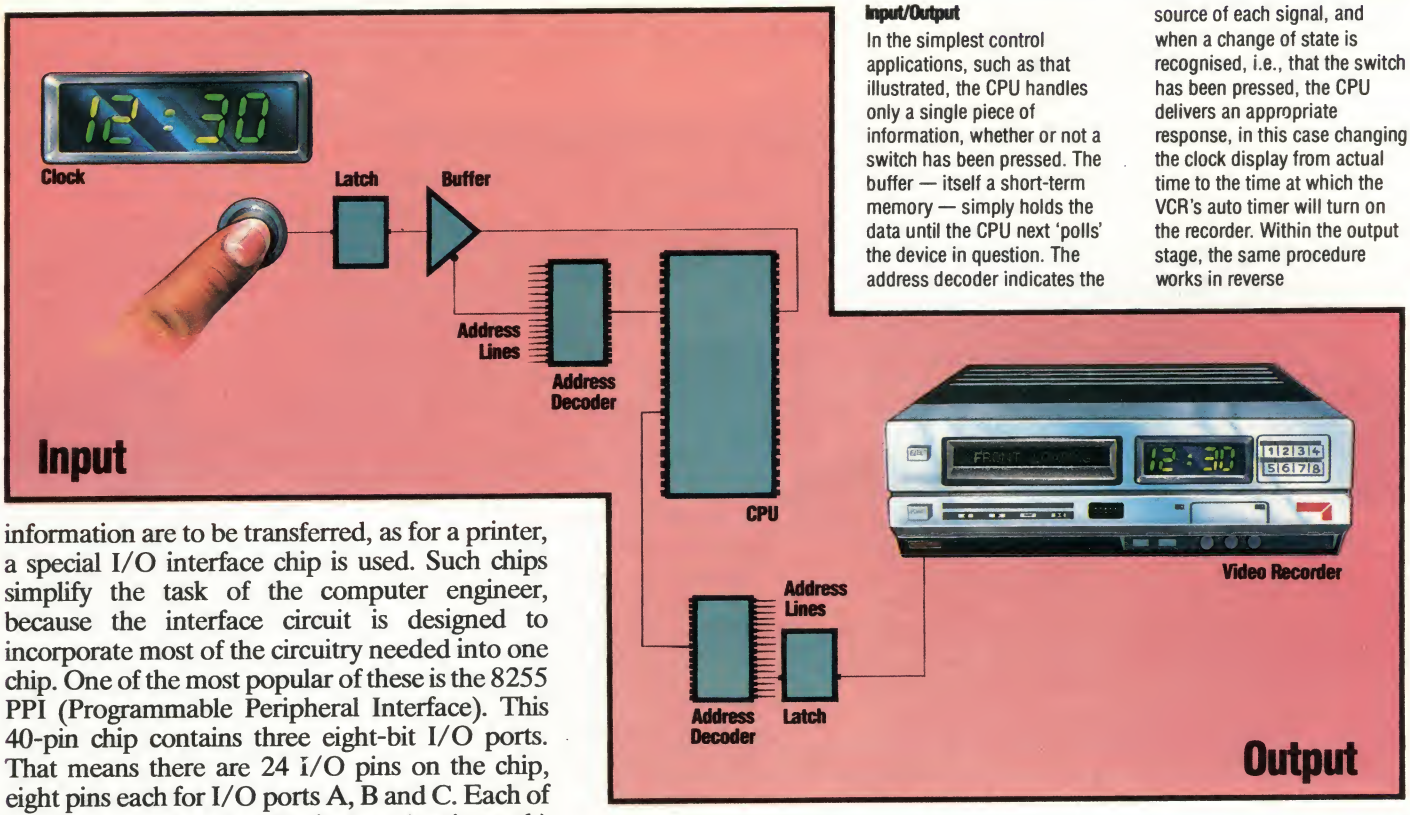
When the CPU wants to retrieve data from memory, it has first to 'address' the location where the byte of data is stored. Similarly, if the CPU wants to store a byte of data for later use, it must first address the location where the item of data is to be stored. This process is called 'memory addressing'. It involves the CPU putting the binary digits corresponding to the desired memory location on a set of 16 wires connected to the CPU's 'address pins'. These wires are called the 'address bus'. Special circuitry in the memory section is able to decode these 16 binary digits to select the correct memory location. (Sixteen binary digits can give 65,536 unique combinations of ones and zeros and can therefore address that many different memory locations.)

If the computer wants to communicate with an external device, it also has to 'locate' that device in a similar way. Only eight address lines are available. This limits the total number of separate I/O locations that can be selected to 256. This is a small number compared with the addressing power of 16 address lines, but in practice 256 is more than adequate. There's usually no need for huge numbers of external devices to be connected to a computer.

Selecting Devices

To find out how the computer actually selects an external device and sends data to it, let's consider one of the simplest output devices possible — an LED (Light Emitting Diode) mounted on the keyboard of the computer to show when the 'caps lock' key has been pressed (there's a key and an LED like this on the BBC Microcomputer). To the computer, the LED is simply another external device to which it can send data. In the case of a single LED, the data will be either a single 1 (to turn the LED on) or a single 0 (to turn the LED off). Even though it is just a humble LED requiring a single bit of data, it still needs to have an address or location. The CPU can't spend its whole time addressing one LED. It needs to be able to select the LED once only to tell it when it needs to switch on, and again to tell it when to switch off. Suppose, for the sake of argument, the LED has an I/O address of 32. To select it, the address lines will have to be set by the CPU to the binary equivalent of 32. This is 00100000 in binary. The LED will have a special 'decoder' circuit that will ignore all other combinations of bits on the address lines. When the address line becomes 00100000, this decoder circuit recognises it and produces a high voltage and therefore a 'true' output. The next part of the circuit needed to switch on the LED is a small chip called a 'data latch'. This latches or holds the data sent to it so that the LED stays on or off until the next time it is addressed and new data is sent to it. This process is known as 'toggling'.

Most of the external devices with which the computer communicates are considerably more complex than a single LED. A printer is a typical peripheral and each time the computer communicates with it, the data transmitted will represent the code for a whole character to be printed. Usually, when large amounts of



Input/Output
In the simplest control applications, such as that illustrated, the CPU handles only a single piece of information, whether or not a switch has been pressed. The buffer — itself a short-term memory — simply holds the data until the CPU next 'polls' the device in question. The address decoder indicates the source of each signal, and when a change of state is recognised, i.e., that the switch has been pressed, the CPU delivers an appropriate response, in this case changing the clock display from actual time to the time at which the VCR's auto timer will turn on the recorder. Within the output stage, the same procedure works in reverse

information are to be transferred, as for a printer, a special I/O interface chip is used. Such chips simplify the task of the computer engineer, because the interface circuit is designed to incorporate most of the circuitry needed into one chip. One of the most popular of these is the 8255 PPI (Programmable Peripheral Interface). This 40-pin chip contains three eight-bit I/O ports. That means there are 24 I/O pins on the chip, eight pins each for I/O ports A, B and C. Each of these ports can send eight bits (one byte's worth) of data at a time to a peripheral device such as a printer, or receive eight bits of data at a time from an input device such as a keyboard.

To send eight bits of data to a printer, the CPU will first address the PPI and then send it the eight bits of data on the data bus. This data will be stored in a temporary one-byte memory cell within the chip, called a register. The PPI will then make this data available on the appropriate set of I/O pins. A similar principle, but working in reverse, allows data from external input devices to be stored in a register in the chip, and then put onto the data bus when the CPU sends it the appropriate signal. As noted above, external devices cannot be allowed to put their data onto the computer's data bus continuously — it is needed to transfer data to and from memory and by other I/O devices. The I/O chip stores the data temporarily and only puts this data on the data bus (to be picked up by the CPU) when the CPU tells it to do so.

How does the CPU know if an external device is trying to send data to the computer? Briefly, there are two main techniques that can be used. The CPU stops running the program it is executing periodically and takes a quick look at all the input ports. If it finds data there waiting to be input, it instructs the port to put the data on the data bus. The process of enquiry of the input devices is known as 'polling'.

The other method uses 'interrupts'. The device wanting attention sends an interrupt signal directly to the CPU and this forces the program being executed to stop while the input port is attended to. The advantages and disadvantages of these two methods will be described in more detail later in the course.

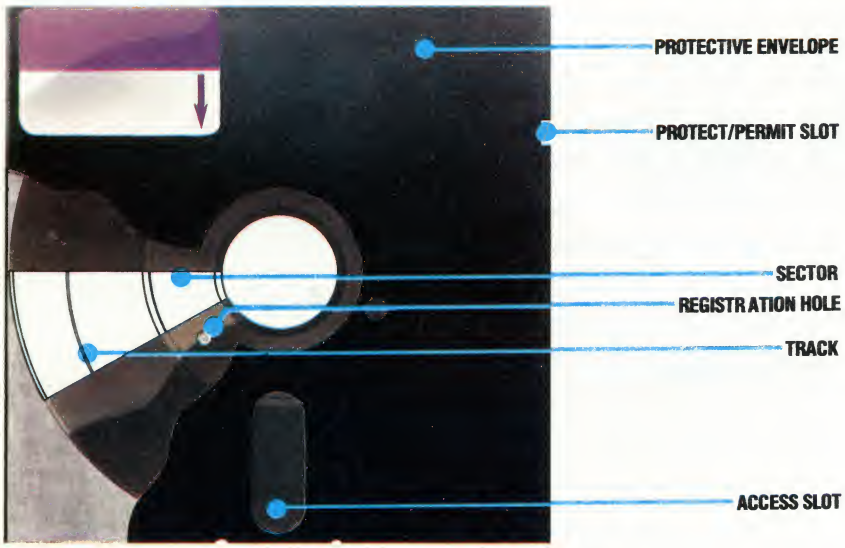
The I/O we have described so far is called 'parallel I/O' because data is input or output one byte at a time using eight I/O wires or lines (eight bits in parallel). Another technique is called 'serial I/O'. Here the information in each byte is fed in or out a bit at a time, one bit after the other. Some printers use serial interfaces, and the output from modems (see page 108) is also in serial form. The main advantage is that, essentially, serial communication allows a single pair of wires to be used instead of eight or more.

Serial And Parallel Ports
Most modern microcomputers provide both serial and parallel ports, the former passing data a bit at a time, the latter in whole bytes. The most common type of serial convention, known as RS232C, uses either a 'D-type' sub-miniature connector, a 25-pin example of which is illustrated here (left), or more rarely a DIN plug like those used in hi-fi systems.

The parallel port (right) follows the IEEE488 convention, developed by Hewlett Packard and adopted as an industry standard by the Institute of Electrical and Electronics Engineers in the USA

Flat Spin

Magnetic disks spin at high speed, within disk drives, carrying information that can be 'read' by your computer



DAVID WEEKS

The Floppy Disk
The surface of a disk is divided up into a number of separate bands called tracks. These tracks are further subdivided into sectors. On the Apple II, for example, each track is divided into 16 sectors. Each sector has an address field and a data field.

The Disk Operating System accesses the individual sectors on a track by using the address field, which contains the track and the sector numbers, and an identifier (to check that the user is reading the right disk). Thus it can retrieve information in much the same way as it is retrieved from a memory location (by using its address)

Home computers will 'forget' everything you have programmed them with once the power is switched off. At best this can be a minor irritation, at worst a major disaster as an entire evening's programming disappears for good. For this very reason, the makers of home computers incorporate a method by which the contents of the computer's memory can be permanently stored. This usually takes the form of a cassette tape on which the program is stored digitally as a series of tones (see page 94).

However, when dealing with long programs, or a collection of small programs that need to be frequently used, the time taken to find and load the program from a cassette can be a major setback. There are two reasons for this. This first is that a tape must be started at the beginning in order to locate a program recorded on it – although cassette recorders with tape counters greatly assist here.

The second cause of the problem is the way in which the program is stored. The patterns of bits held in the memory have to be converted into a corresponding sequence of tones: a high tone represents a bit that is on (or set to one), and the lower tone represents a bit that is off (or set to zero). These tones must then be recorded onto the cassette tape. The fastest practical rate at which this transfer can occur is 150 bytes a second. Any faster, and the possibility of errors increases to the point where the system fails to be reliable.

A conventional cassette system using C-10 tape can take as long as five minutes on each side to find and locate a program. This is assuming that a fast loading system is being used. Some systems work as slowly as 30 bytes a second. For those long programs you really need a recording system that will find the beginning of the program and load it in a matter of seconds.

Such a storage system is the floppy disk and it can be used on most of today's home computers. If you imagine the yards of tape stored inside a cassette tape laid out in the form of a spinning disk some five inches across, you will appreciate how quickly any information stored on the disk can be located. This disk is placed inside a protective envelope and slotted inside a disk drive.

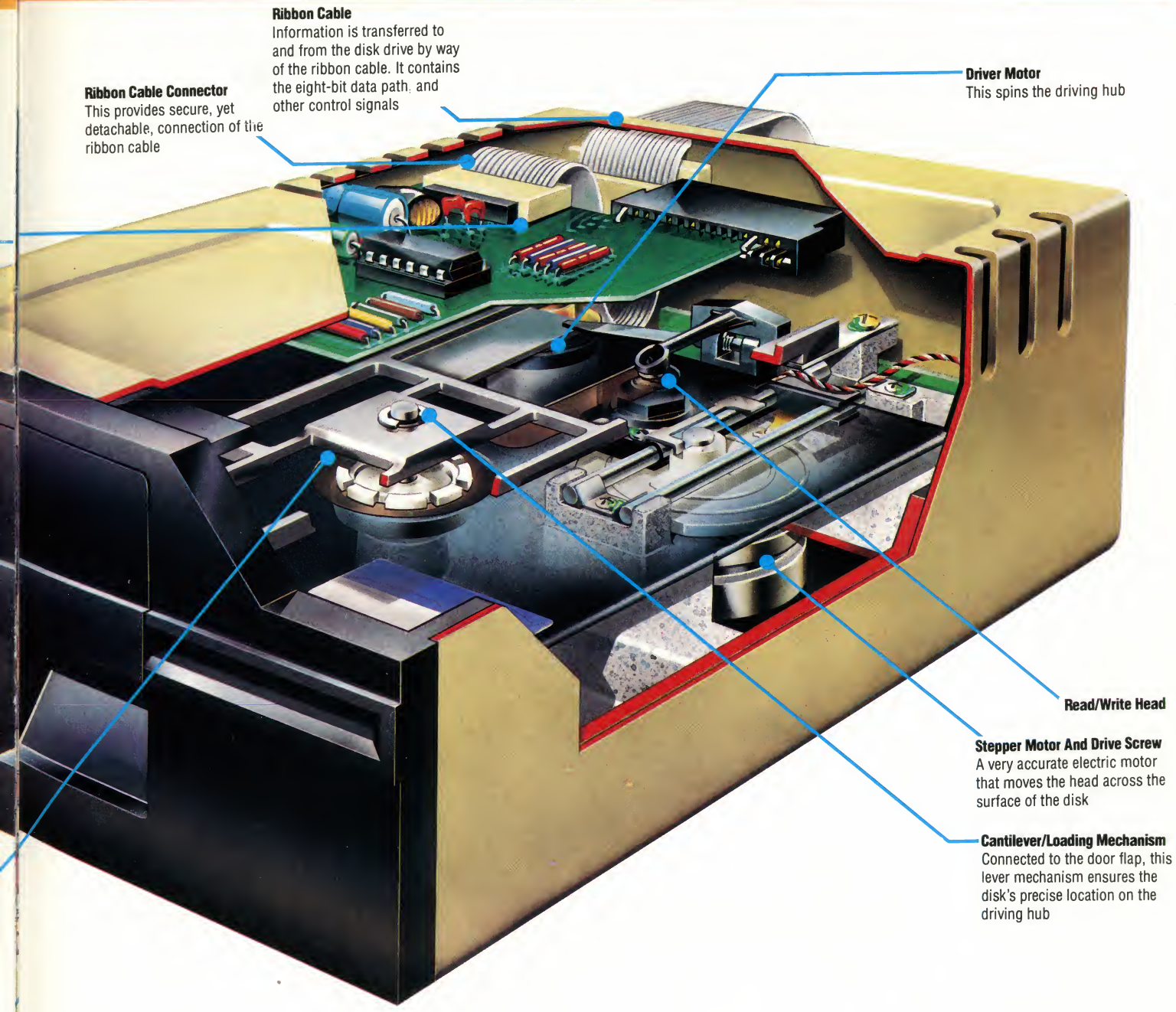
The drive's function is to spin the disk (inside its envelope) at a constant speed, and to provide a means of transferring programs on and off the disk from the computer. It does this through a recording and playback head, similar to that on a cassette recorder, but very much smaller. This head can move backwards and forwards across the surface of the spinning disk, unlike the cassette, which can only move the tape past the head.

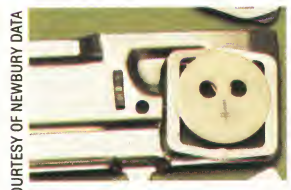
Analogue Board
This circuitry converts the signals coming from or going to the head. It translates the digital form used in the machine to the analogue form that goes on the disk

Indicator
This Light Emitting Diode shows whether access is being made to the disk drive

Driving Hub
This engages with the plastic disk and spins it round inside the envelope

Care Of Your Disk
Floppy disks are delicate and should be handled with respect. Follow the manufacturer's recommendations carefully


Ribbon Cable
Information is transferred to and from the disk drive by way of the ribbon cable. It contains the eight-bit data path, and other control signals
Ribbon Cable Connector
This provides secure, yet detachable, connection of the ribbon cable
Driver Motor
This spins the driving hub
Read/Write Head
Stepper Motor And Drive Screw
A very accurate electric motor that moves the head across the surface of the disk
Cantilever/Loading Mechanism
Connected to the door flap, this lever mechanism ensures the disk's precise location on the driving hub



COURTESY OF NEWBURY DATA

Read/Write Head
This is a highly magnified picture of the head that reads and writes data to the surface of the disk. It is similar to the head on a cassette recorder, but almost invisible to the naked eye.

Unlike a tape that is just one long string of bytes, a disk is 'formatted' in a series of concentric circles, each of which is treated by the system as small chunks, usually 256 bytes each. Each of these 'sectors' has an address.

When a program is to be written to the disk, the first thing that happens is that the head is moved to the directory, a special file which acts as an index to the whole disk. This is examined to find out where to put the file. If it's being re-written, the first sector of the old copy is found, and the new data is stored starting there. A new file won't have an entry in the directory, so one must be made, then the first empty sector is filled with the data, with more sectors being filled as required.

The advantages of high-speed efficiency and large storage capacity offered by the disk explain the substantial difference in price between the two systems. Disk drives sell from about £120, whereas a cassette player is usually less than £20.

The most important factor in this price difference is the precise engineering that is required. The recording and playback head of a disk drive is almost invisible and must be placed to within hundredths of an inch.

The mechanism that moves this minute head is based on an electric motor, which can turn by fractions of a degree. This is coupled to a shaft that carries the head and moves it across the surface of the disk in minutely calculated steps. To ensure that the disk spins at a constant speed, complex eléctronics are used and all the components are mounted on a rugged die-cast frame to reduce the effects of heat and vibration.

Braving The Elements

Subscripted variables, unlike their simple counterparts, can contain any number of elements

In our earlier program for calculating the number of days to Christmas we encountered a new type of variable called a 'subscripted' variable. These differ from ordinary or 'simple' variables in that they can have any number of compartments or elements within the box. Simple variables recognise two letters or letters followed by a digit from 0 to 9 (some versions of BASIC allow whole words to be used as variable names). A, B, B1, C3 and R2 are all simple variables. Subscripted variables look like this: A(6), B(12) or X(20). The subscript is the number in brackets. The examples we have given would be read as: 'A sub six', 'B sub twelve' and 'X sub twenty'.

If we think of a simple variable as being a box with a name or label on it, we can think of a subscripted variable as a box containing a specified number of internal elements. If we want a variable with 12 elements, we create it initially using the BASIC DIM statement, like this: DIM A(12). Any letter of the alphabet may be used.

Assigning values to simple variables is straightforward, using either LET or INPUT statements, like LET A = 35, LET B1 = 365 or INPUT C3. Values can be entered in the elements of a subscripted variable in the same way. Let's see how we would assign values to a subscripted array. ('Array' is an alternative name for a set of subscripted variables.) For example:

```
10 DIM A(5)
```

creates a subscripted variable with five elements. We can now assign a value to each element:

```
20 LET A(1) = 5
30 LET A(2) = 10
40 LET A(3) = 15
50 LET A(4) = 20
60 LET A(5) = 100
```

To find out how these variables differ from simple variables, let's assign values to a few simple variables:

```
70 LET X = 5
80 LET Y = 6
90 LET Z = 7
```

Try entering all these on your computer and then check the contents of each variable using the PRINT command. Many of the statements in BASIC also function as commands. After you have entered the statements above, check them by LISTing them and then type RUN. Now you can type PRINT X<CR>. You should see 5 instantly displayed on the screen. Next type PRINT Y. The computer will respond to this PRINT command by displaying 6 on the screen. If you want to check the elements in the subscripted variable, type PRINT A(1) to find out the value of the first element in the array. The computer should respond by printing 5 on the screen. Try PRINTing the values of A(3) and A(5).

The important difference between subscripted variables and ordinary variables is that the subscript can itself be a variable. To see what this means, type PRINT A(X). The screen will respond with the figure 100. Why?

Look at the list you have typed in and check the value of variable X. It is 5. A(X) is equivalent to A (the value of variable X) and this is equivalent to A(5). Typing PRINT A(X) is therefore exactly equivalent to typing PRINT A(5). What value would you expect if you typed PRINT A(Y — X)? Before actually trying it, see if you can work out the answer.

Assigning Values

If there are only a few simple variables, the LET statement is the simplest way of assigning values to them. Subscripted variables may well have a large number of elements in the array, so let's see what the alternative methods of entering the values are:

```
10 DIM A(5)
20 PRINT "INPUT THE VARIABLES"
30 INPUT A(1)
40 INPUT A(2)
50 INPUT A(3)
60 INPUT A(4)
70 INPUT A(5)
```

This method is just as tiresome to type in as using LET statements, though it would certainly work. If we know exactly how many variables there are (in this case there are five) it is easier to use a FOR-NEXT loop, like this:

```
10 DIM A(5)
20 FOR X = 1 TO 5
30 INPUT A(X)
40 NEXT X
```

This program would expect five values to be typed on the computer keyboard when the program was run. The RETURN key would have to be pressed after each figure had been entered. If we know beforehand what the values in the variable are, it is

easier to enter them using a READ statement together with a DATA statement, like this:

```
10 DIM A(5)
20 FOR X = 1 TO 5
30 READ A(X)
40 NEXT X
50 DATA 5, 10, 15, 20, 100
```

Try this short program, and then test the contents of the array using the PRINT command (that is, use PRINT after the program has been RUN. For example, PRINT A(1)‹CR› and PRINT A(5). Now we can add a few lines to the program to print the elements in the array for us automatically:

```
60 FOR L = 1 TO 5
70 PRINT A(L)
80 NEXT L
90 END
```

RUN this program and check that the correct values are printed on the screen. Then retype line 50 using five different DATA items. Remember that the numbers in a DATA statement must be separated from each other using commas, but there must be no comma before the first number or after the last one.

The simplest way to assign values is to use READ and DATA statements. If the values will be different every time the program is run, using the INPUT statement inside a FOR-NEXT loop is probably the best way. If the total number of elements in the array is fixed, the number can be used as the upper limit in the FOR statement.

Let's use all we have learnt so far to build a short but powerful program. Suppose we wanted to sort some numbers into ascending order. Before setting out to write the program, the first thing to do is to figure out how to solve the problem in a logical way. When the way to solve the problem seems clear, write down the steps one after the other using clear, short English sentences.

Suppose we start with five numbers: 4, 9, 2, 8, 3. Sorting these into ascending order is a trivial problem. We just scan along the line and notice which is the smallest, and put it on the left, and then repeat the process for the remaining digits.

The computer, however, needs a very precise set of instructions, so we shall have to think very clearly about what steps are required. Here's one approach: Compare the first digit with the second digit. If the first digit is bigger than the second one, swap them. If the first digit is smaller than the second one, leave their positions unchanged.

Compare the second digit with the third digit. If the second digit is smaller than the third one, leave their position unchanged.

Repeat the process of comparing pairs of digits until the last pair of digits has been compared.

If there were no swaps, all the numbers must be in order. If there were any swaps, go back to the beginning and repeat the process.

If you think about this process, you will see that it will indeed sort any group of numbers into ascending numeric order. Look at what would happen to our original set of numbers as each pair of digits is compared:

```
4  9  2  8  3
4  2  9  8  3
4  2  8  9  3
4  2  8  3  9
```

All the pairs have now been compared and swapped where necessary. Since at least one swap took place, go back to the beginning and repeat the process:

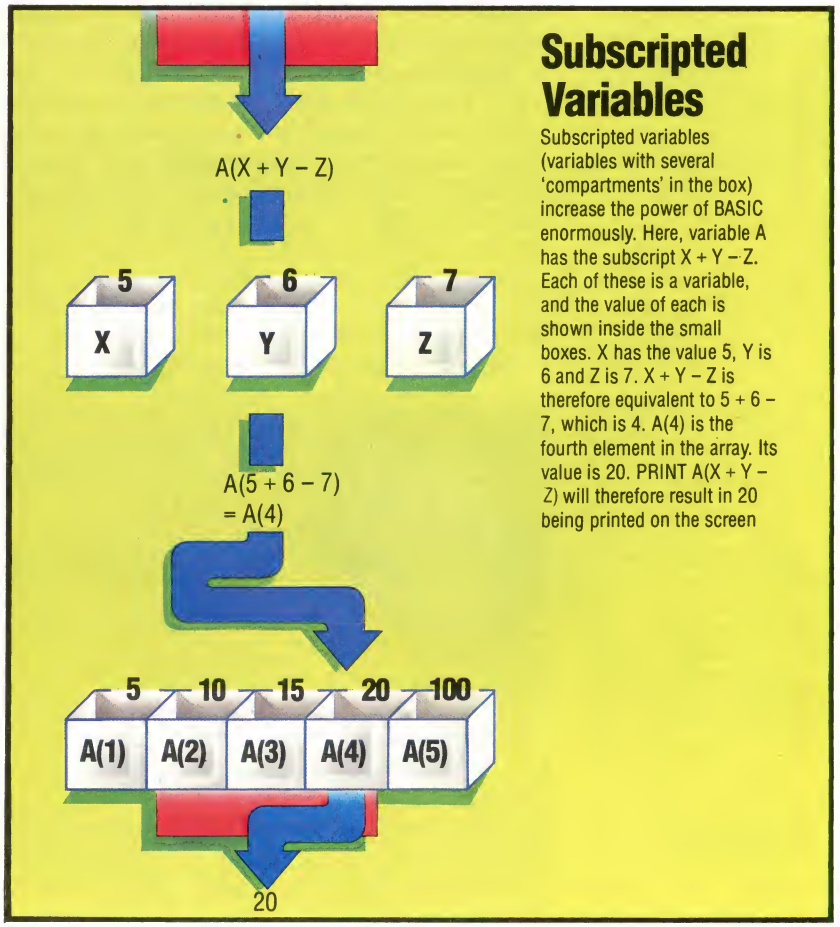
```
4  2  8  3  9
2  4  8  3  9
2  4  3  8  9
2  4  3  8  9
```

There were still swaps, so go back to the beginning and repeat:

```
2  4  3  8  9
2  3  4  8  9
2  3  4  8  9
```

There were no swaps, last time through, so every number must be smaller than the number to its right. The numbers must be in ascending order and the operation can be terminated.

Using subscripted variables allows a sort routine like this to be implemented easily in BASIC, because the subscript itself can be a variable. If our original five numbers were the values in an array; so that A(1) = 4, A(2) = 9, A(3) = 2, A(4) = 8 and



Subscripted Variables

Subscripted variables (variables with several 'compartments' in the box) increase the power of BASIC enormously. Here, variable A has the subscript X + Y – Z. Each of these is a variable, and the value of each is shown inside the small boxes. X has the value 5, Y is 6 and Z is 7. X + Y – Z is therefore equivalent to 5 + 6 – 7, which is 4. A(4) is the fourth element in the array. Its value is 20. PRINT A(X + Y – Z) will therefore result in 20 being printed on the screen

TONY LODGE

A(5) = 3, then if X has the value 1, A(X) will be the contents of A(1), which is 4. A(X + 1) will be the contents of A(2), which is 9, and so on.

Look at the program and see if you can see exactly what is going on. Line 20 sets variable N to the number of numbers we want to sort. Let's assume we want to sort five numbers: when the program is run we will type in 5 and then hit RETURN.

Line 30 is the DIMension statement. If N is 5, it sets the size of the array to 5. This line is equivalent to DIM A(5).

Lines 40 to 60 are a FOR-NEXT loop that allows us to type in the five numbers. Most versions of BASIC prompt the user with a question mark on the screen. RETURN will have to be pressed after each number has been entered. The numbers may be more than one digit, and may include decimal fractions.

Line 90 sets the variable S to 0. This variable is being used as a 'flag'. Later in the program, A is tested to see if it is 1 or not. It is only ever set to 1 if two numbers have been swapped, as we shall see in line 240. We shall investigate the use of 'flags' in more detail later in the course.

Line 100 sets up the limits for a loop; in this case from 1 to 4 (because N is 5 so N — 1 is 4). The first time through the loop, L is 1 so A(L) in line 110 will be A(1) or the first element in the array and A(L + 1) will be A(2), the second element in the array. The next time round the loop, L will be incremented to 2, so A(L) will be equivalent to A(2) and A(L + 1) will be equivalent to A(3). Line 110 tests to see if A(L) is greater than the number immediately to its right in the array. The sign for 'greater than' is ›.

If the first number is bigger than the next one, the program branches to a subroutine that swaps the numbers. If the first number is not bigger than the next one, there is no branch to the subroutine and BASIC simply continues to the next line, which is the NEXT L statement. After the loop has been repeated four times, it stops the program and goes to line 130 which tests the 'swap' flag, S, to see if it has been set or not. If it has been set (in the 'swap' subroutine), the program branches back to line 90 to repeat the comparison process. If S is not 1 it means no swap took place, so all the numbers are in order. The rest of the program simply prints them out.

The swap subroutine needs a temporary variable to store one of the numbers to be swapped. After the two numbers have been swapped in lines 210, 220 and 230, the 'swap' flag S is set to 1 and then the program RETURNs to the main program.

```
10 PRINT "HOW MANY NUMBERS DO YOU WANT
TO SORT?"
20 INPUT N
30 DIM A(N)
40 FOR X = 1 TO N
45 PRINT "NEXT NUMBER"
50 INPUT A(X)
60 NEXT X
70 REM
80 REM SORT ROUTINE
90 LET S = 0
100 FOR L = 1 TO N — 1
110 IF A(L) › A(L + 1) THEN GOSUB 200
120 NEXT L
130 IF S = 1 THEN 90
140 FOR X = 1 TO N
150 PRINT "A(";X;") = ";A(X)
160 NEXT X
170 END
180 REM
190 REM
200 REM SWAP SUBROUTINE
210 LET T = A(L)
220 LET A(L) = A(L + 1)
230 LET A(L + 1) = T
240 LET S =1
250 RETURN
```

Exercises

■ Extend the program to find the average value of the numbers input. The average is equal to the sum of items divided by the total number of items. The simplest way to do this is to insert a GOSUB just before the END statement in line 170. The subroutine should read each of the elements in the array and add the values to a 'sum' variable. After all the elements have been added, the sum should be divided by the number of elements. The sum is most easily derived by using the number of elements as the upper limit of a FOR-NEXT loop.

■ Change one line in the program so that the numbers will be sorted in descending order.

■ This exercise is directed mainly at owners of the TI99/04A which does not like having variables used as subscripts in subscripted variables. TI BASIC does, however, accept statements such as DIM A(12). Rewrite the program so that the INPUT statement expects an exact number of numbers to be input, 12, say. This will avoid the problem of using a variable name as a subscript. Lines 100 and 110 will also have to be changed. The swap subroutine will not work in TI BASIC for the same reason. This will have to be changed too.

■ A tough one. Our way of sorting numbers is by no means the only way to do it. See if you can think up an alternative method.

Basic Flavours

IF... THEN — If this program is to be run on the ZX81 or Spectrum, line 130 must be amended to read: 130 IF S=1 THEN GOTO 90

END — This statement is not available on the ZX81 and Spectrum, so use STOP instead

LET — In assignment statements, such as 90 LET S= 0, the word LET is optional on most machines but not on the ZX81 and Spectrum. The Lynx adds it into program listings automatically

Two's Company

Although computers give speedy answers to complex arithmetical problems, they deal with them in the simplest way

In the last part of our course on binary, we discovered how computers could add. Now we look at the process of multiplication.

If you had to multiply 14 by 12, a simple way would be to do a multiple addition, for example 14+14+14+14+ . . . (12 times). Since multiplication is in one sense a form of repeated addition, this approach would certainly work, and is how the first computers dealt with multiplication. However, the method is awkward and time-consuming, so computer designers evolved a more efficient method.

When two numbers are multiplied, the operation is usually written down on paper like this:

```
  14
 ×12
  28
 +14      (a final 0 is often written
 168       to keep the digits in
           the correct column)
```

Exactly the same process works in any base of numbers. Let's look at an example in base two or binary:

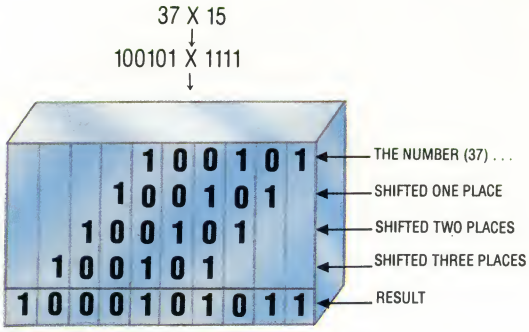
```
  101
 ×11
  101
 +101
 1111
```

With larger numbers the method is exactly the same, so let's go back to the example of 14 × 12 and do it in binary:

```
    1110     (14)
   ×1100     (12)
    0000
   0000
  1110
 1110
10101000     (168)
```

Multiplication is even simpler in binary than in decimal because there can never be a carried digit. When you multiply a number by 1 the number is unchanged, 14 × 1 = 14 and when you multiply a number by 0 the answer is 0, 14 × 0 = 0. This is true in binary, decimal and all number systems.

When mathematicians looked at similar calculations to the one above they saw a simple pattern emerging: binary multiplication consists of only two operations, 'shifting' and addition. And this is exactly how the computer does a multiplication. First it shifts 'copies' of the upper line into their correct position (determined by the 1s and 0s in the lower line) and then it adds all the 'copies' together.

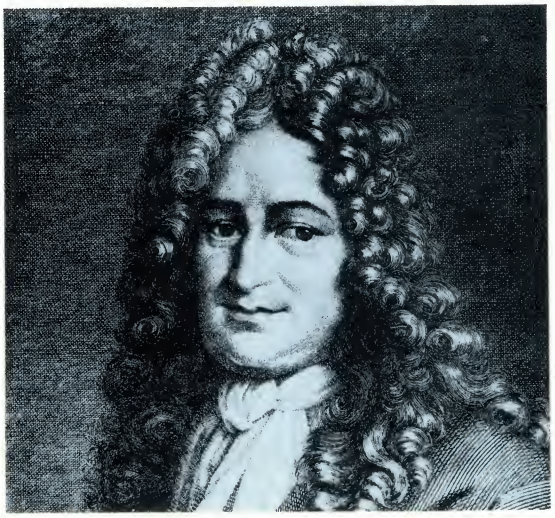

37 X 15
100101 X 1111
1 0 0 1 0 1
THE NUMBER (37) . . .
1 0 0 1 0 1
SHIFTED ONE PLACE
1 0 0 1 0 1
SHIFTED TWO PLACES
1 0 0 1 0 1
SHIFTED THREE PLACES
1 0 0 0 1 0 1 0 1 1
RESULT

KEVIN JONES

Shifting Into Multiplication
Multiplication is much easier in binary than it is in decimal. The same process of long multiplication used in the decimal system is applied, but because there are only two numbers involved in the multiplying (0 and 1), the operation is very simple. When a number is multiplied by 1 the result is obviously the same number. In the illustration, where 100101 (37) is multiplied by 1111 (15), four copies of 100101 appear. Each copy is shifted to the left according to the position of the 1 that multiplies it. Finally all the copies are added together to give the answer 1000101011 (555)

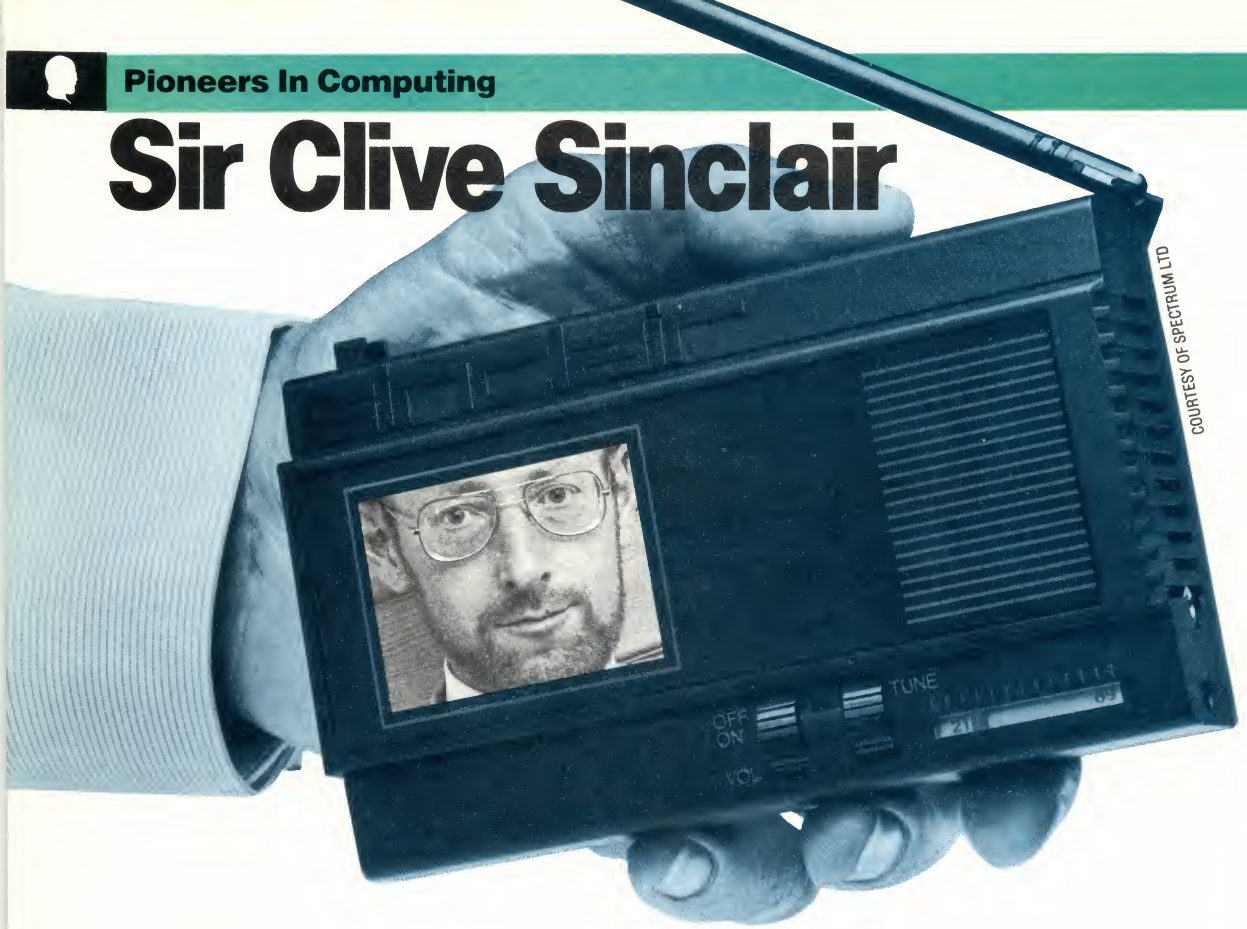
The computer needs to have a large digit capacity to perform multiplications. When the 4-digit number 1110 was multiplied by the 4-digit number 1100 the answer contained 8 digits (10101000) and in general the result of a multiplication can be up to twice the length of the largest number.

It may come as a surprise to learn that a computer's multiplication can return an incorrect result. Nearly all these errors can be traced back to the amount of space the machine's designer has allowed to hold the answer. If insufficient space has been allocated, 'overflow' will occur, the least significant digits will be lost, and the result will thus be erroneous.



Gottfried Wilhelm Leibniz
Leibniz (1646–1716) was a contemporary of Isaac Newton and made contributions to mathematics, science and philosophy. He invented a machine that could multiply and divide. He also investigated the possibilities for using binary arithmetic in calculating devices, though the first known reference to binary numbers was by Francis Bacon in 1623. In his later years he fell into dispute with Newton over the invention of the calculus

Sir Clive Sinclair



The entrepreneur/engineer who made the computer more widely available

Stop someone in the street and ask them if they can describe what a Spectrum or a ZX81 looks like and you'll probably get a blank stare. But ask if they have heard of Sir Clive Sinclair and the chances are anybody who reads newspapers and watches television will know that you are talking about a millionaire electronics genius.

It is fair to say that this 43-year-old ex-technical journalist is the most famous individual in the computer world. He has been described as doing for the personal computer what Henry Ford did for the motor car and Freddy Laker achieved in the air travel industry.

His success at designing and marketing the world's most successful computer — the ZX81 — was rewarded earlier this year when he was awarded a knighthood for putting Britain back in the technological race against Japan and the USA.

Clive Sinclair was born in London in 1940. He left school at 17 after completing his education at St George's College in Weybridge. He was a technical journalist for four years before forming his first company, Sinclair Radionics, in 1962. The first products were radio and amplifier kits sold by mail order.

Sir Clive has made his fortune from his ingenious computer designs that have turned into the ZX81 and ZX Spectrum. But he was also responsible for the world's first low-cost, electronic calculator which he launched in 1972. There was even a gold plated model which sold for £2,750! He also made one of the first digital wristwatches using a microchip.

His most recent innovation is the flat-screen television which is no bigger than a paper-back book and which has a two-inch screen. This will sell for £80 and will be available later this year.

Now Sir Clive is working on his most ambitious project of all — an electrically-powered car. The team working on this project is working towards building a new electric vehicle for town use.

But Sir Clive is also an established publisher. He has written around 17 books himself on electronic subjects and in 1981 launched a new publishing company called Sinclair Browne. He publishes around 20 fiction and non-fiction books a year.

Early in 1983 the *Guardian* newspaper named him 'Young Businessman Of The Year'. The influential trade magazine *Computing* gave him the title 'Computing's Person Of The Decade'.

He escapes from the world of electronics by going to the theatre and poetry readings and he is a trustee of the Cambridge Symphony Orchestra. Sir Clive divides his time between his Cambridge headquarters and his London office and his silver Porsche 924 Carrera can often be seen streaking along the A1.

Sir Clive is a great patriot. His aim is to persuade fellow Britons that there is nothing that the American and Japanese can do that can't be done in their own country.

1962
Clive Sinclair forms Sinclair Radionics in Islington, London, to sell radio and amplifier kits by mail order

1972
Sinclair produces world's first pocket calculator, the Executive, which sells for £79 and earns over £2.5 million in export revenue

1975
Sinclair launches one of the first digital watches, which he calls the Black Watch. However, the company sustains losses due to difficulties with chip supplies

1976
National Enterprise Board gives Sinclair cash to develop his pocket television which is put onto the market the following year after a 12-year development programme

1979
Sinclair sets up a new company, Sinclair Research, to develop products in the consumer electronics field

1980
The new company launches its first product, the ZX80 computer, which is acclaimed as the first computer to sell for less than £100

1981
Sinclair developes the ZX81 computer, which wins a Design Council Award and sells more than one million units in two years

1982
The Spectrum is introduced to sell alongside the ZX81 but designed for a wider range of home, office, and educational uses

1983
The long-awaited Microdrive arrives, along with the Interface I and Interface II, which expand the Spectrum to take ROM cartridges and local area network facilities. Sinclair also announces his new flat-screen personal television set after a four-year £4 million development programme

1984
Sinclair launches the QL, aimed at the small business market, with 128 Kbytes of RAM and 200 Kbytes of back-up in two built-in Microdrives

Home Computer News

The *Personal Computer World* Show, held last year at the end of September, has become the leading venue for the unveiling of new computer products, and the exchange of ideas between computer owners, clubs and companies. Amongst the crowds of schoolchildren eager to demonstrate their prowess at every new arcade game, bewildered businessmen trying to decide which micro offers the best value, and the cacophony of explosions and dalek-like voice synthesisers, were to be seen a vast array of new computers, add-ons and software. Some were available for sale, though an unfortunate number appeared to be little more than manufacturer's prototypes.

◀ New From Sharp
The Sharp PC-5000 is a truly portable computer complete with an eight- line Liquid Crystal Display and an optional built-in silent printer. It uses 'Bubble Memory', which has approximately the same capacity for storage as a disk drive, but no moving parts

Flight Simulators
The number of games programs and specialist games software companies is growing each month. Several produce Flight Simulators – which are both educational and fun
▼

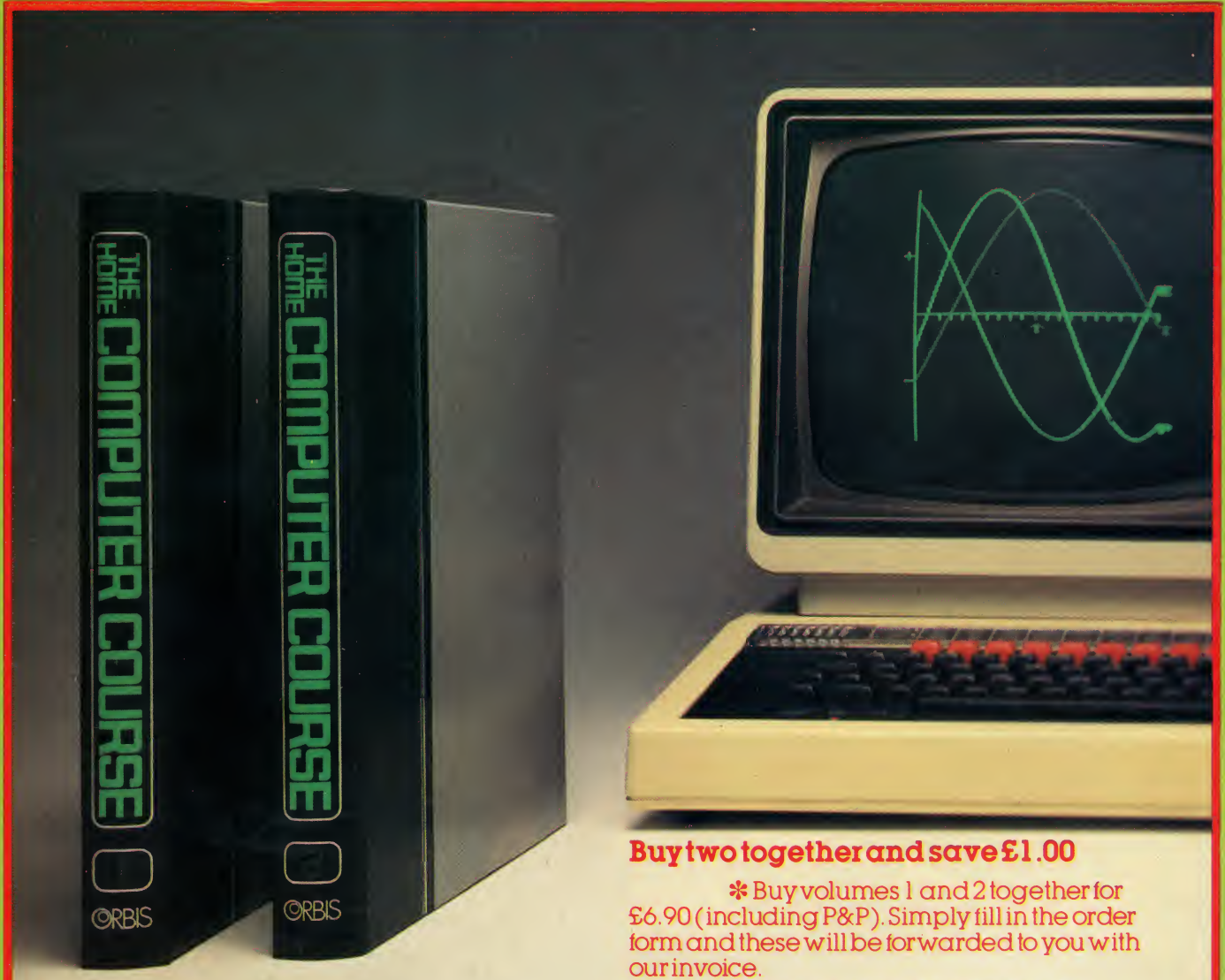
Spectrum Interface ▶
Plug-in cartridge software is now available for the Sinclair Spectrum. But you will need to buy a special interface, which can also connect with joysticks

Talking Back
'My Talking Computer' from Electroplay is designed specifically for teaching young children, and features a talking clock and other educational features ▼

The Buzzbox ▶
A modem is the means of sending data from one microcomputer to another over the telephone. The new Buzzbox from DaCom Systems is not only compact but, at £80, probably the cheapest modem on the market

◀ Three-Dimensional Games
Arcade games have now become three-dimensional, though special glasses are needed to view the effect. The screen displays separate images for the left and right eyes

THE HOME COMPUTER COURSE BINDER



Now that your collection of Home Computer Course is growing, it makes sound sense to take advantage of this opportunity to order the two specially designed Home Computer Course binders.

The binders have been commissioned to store all the issues in this 24 part series.

At the end of the course the two volume binder set will prove invaluable in converting your copies of this unique series into a permanent work of reference.

Buy two together and save £1.00

* Buy volumes 1 and 2 together for £6.90 (including P&P). Simply fill in the order form and these will be forwarded to you with our invoice.

* If you prefer to buy the binders separately please send us your cheque/postal order for £3.95 (including P&P). We will send you volume 1 only. Then you may order volume 2 in the same way – when it suits you!

Overseas readers: This binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in Issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain their binders **now**. For details please see inside the front cover.

Binders may be subject to import duty and/or local tax.

THE LAST WORD IN LOGIC